① Find a loop in a Linked list:

② where in loop:

Node* P = head;

K = Node at(i==j)

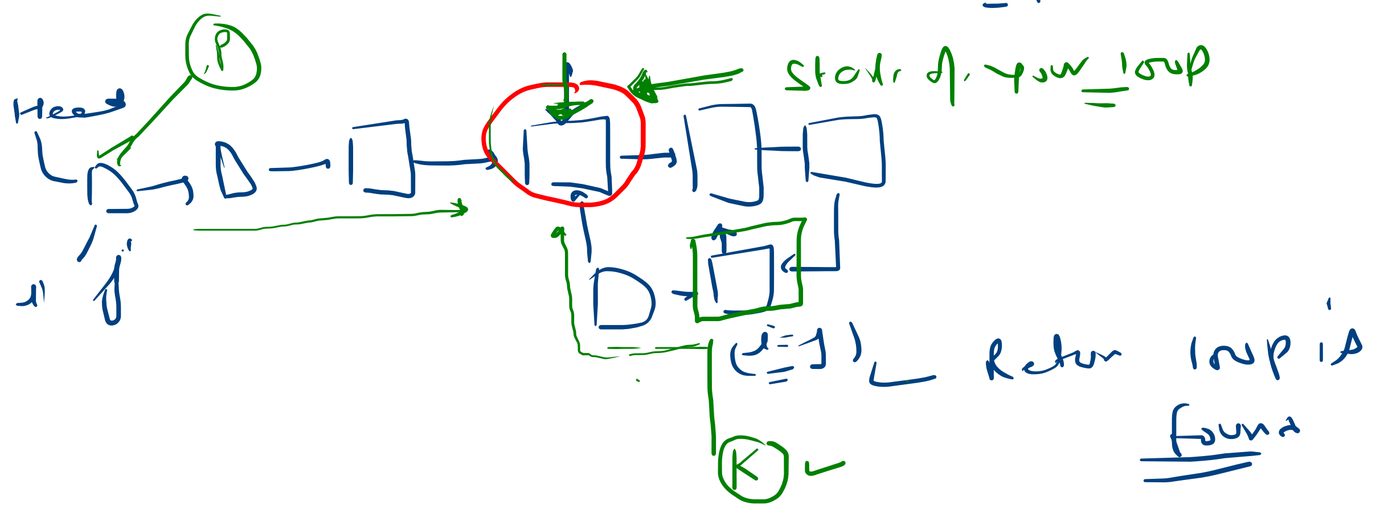
while (P != K)

P = P → Next;

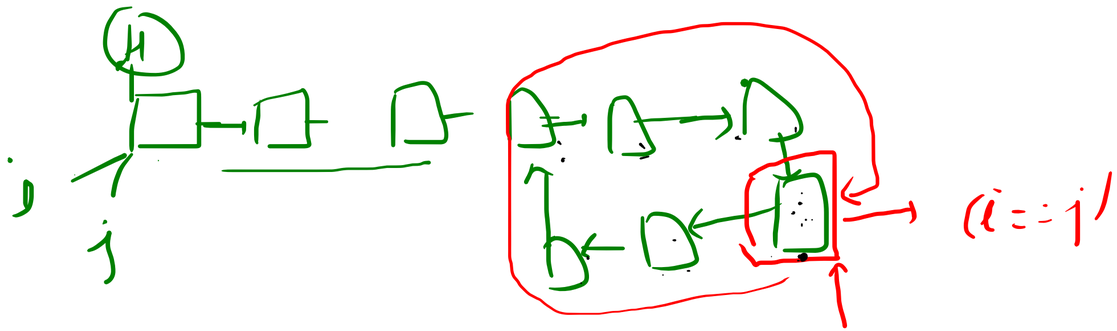
K = K → Next;

(i, j)
+1 +2

⇒ Same (i==j) ⇒ We have found a loop



Loop length:



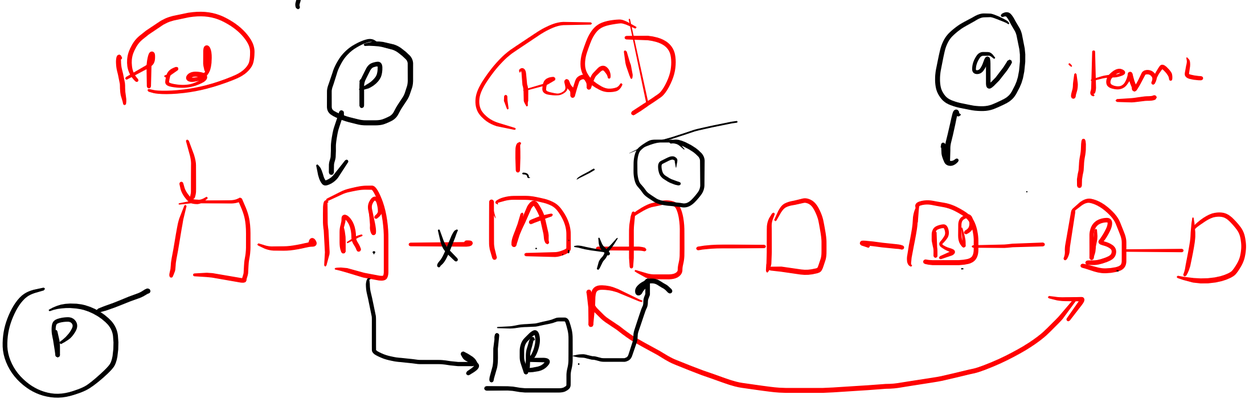

```
Node * start = head;
j = head
i = head.

while ( i != NULL && j != NULL)
{
        i = i -> Next;
        j = j -> Next -> Next;

if (i == j)
{
    Print "Loop found";
    int l = CountNodeLoop (i);
    break;
}
}
Print(
```

```
int CountNodeLoop ( Node * K)
{
Node * start = K;
    Count = 0;
DO {
    K = K -> Next;
    Count ++;
    =
}
while ( K != start)
return (Count);
```

① Swap linked list items:

Swap(Node* head, item1, item2):

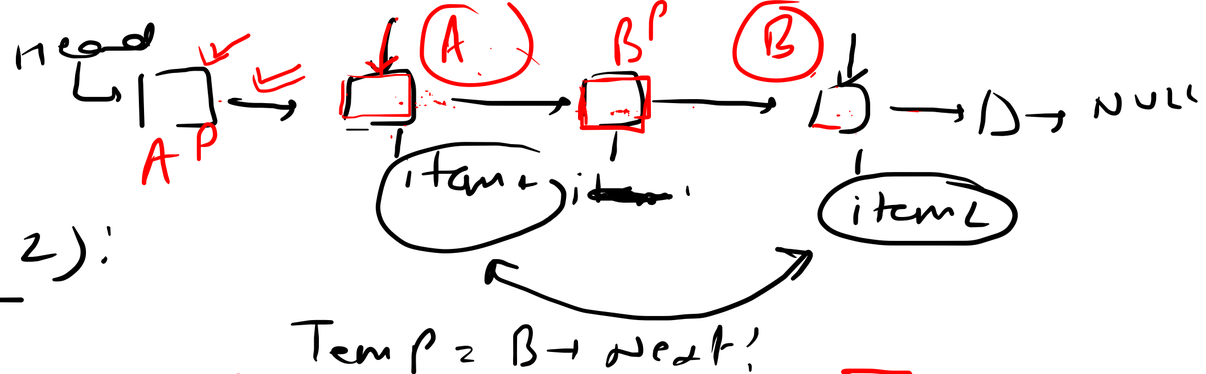
{



Node* P = head; q = head

while (P→Next→item != item②)

P = P→Next;



Temp = B→Next;

A→Next = B ✓

B→Next = A→Next;

B→Next = A;

A→Next = B→Next

Temp = q→Next→Next;

P→Next = q→Next;

q→Next→Next = P→Next→Next

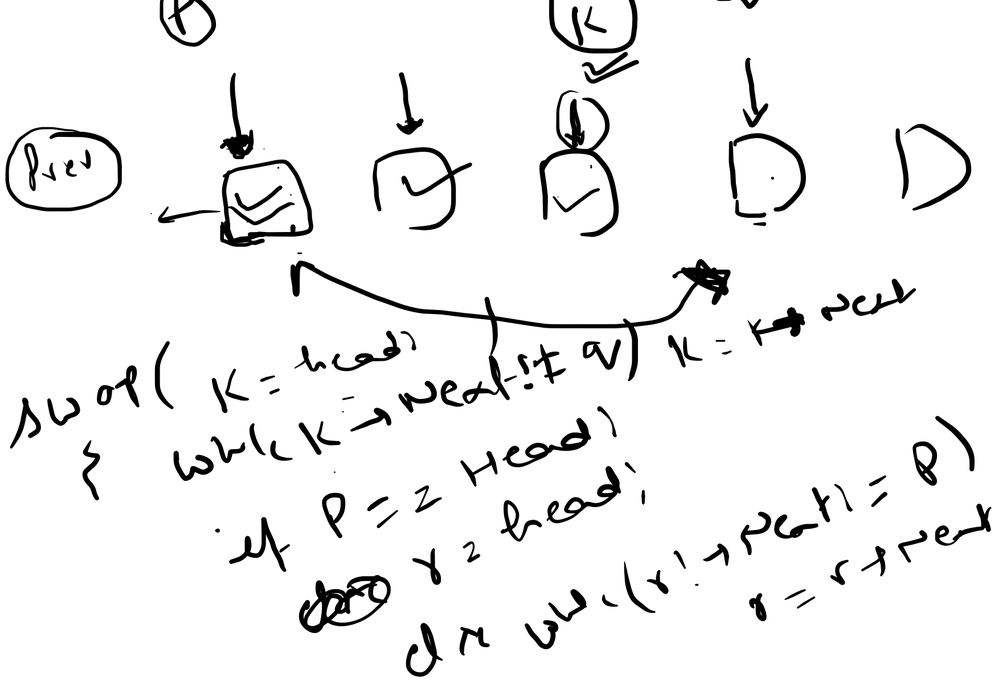
q→Next = P→Next;

P→Next→Next = q→Next→Next;

① Sort linked list:

selection

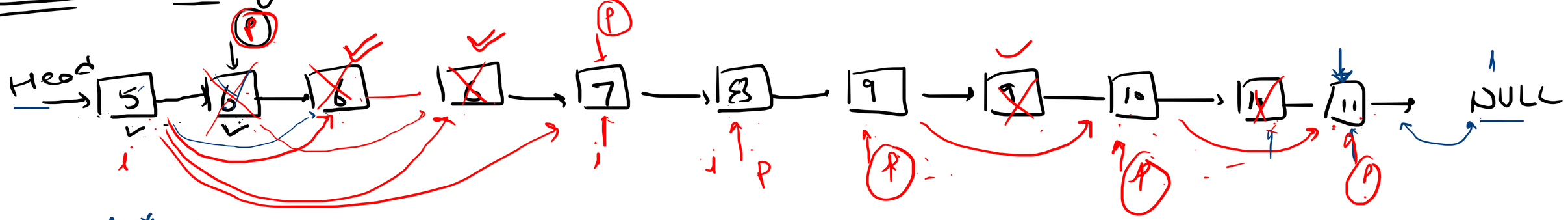
```
for(i=0, i<N ; i++)
for(j=i+1, j<N ; j++)
    if A[j] < A[i]
        swap A[i] A[j]
```




```
swap( K=head)
{ while(K->next!= Q) K = K->next
  if P == Head)
     Y = head;
  do while(Y->next = P)
        Y = Y->next;
```

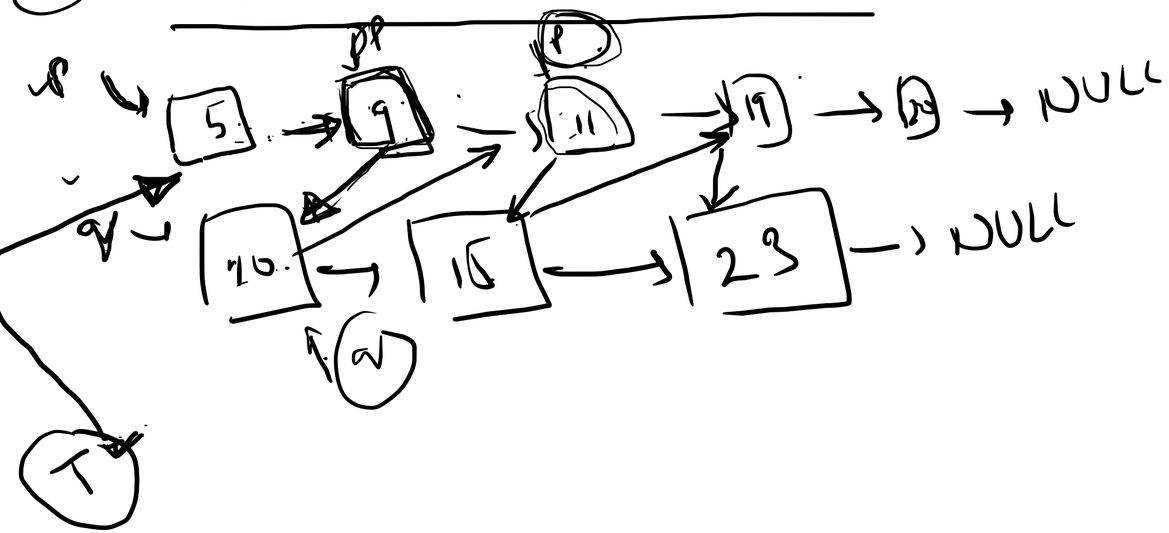
```
P = head
=
while( P != NULL )
 { q = P->Next)
   while( q != NULL)
   {
     if (P->item > q->item)
        swap( P, q, head);
            q = q -> Next;
     P = P -> Next
   }
```

Problem: Finding Duplicate and remove it in sorted linked list.




```
Node* p = head;
While(p->Next != NULL)
{   if (p->item == p->Next.item)
    {   Temp = p->Next
        p->Next = p->Next->Next;
        free(Temp);
    }
else   p = p->Next;
}
```

① Merge Two Sorted linked list:



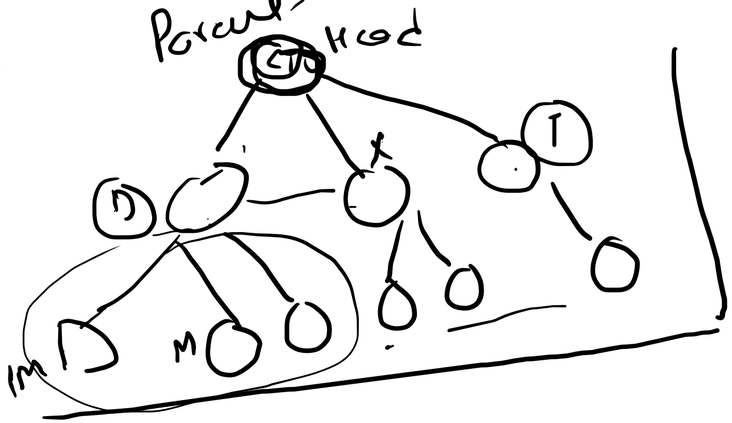
A = 2 3 9 10 15

B = 10 , 12 , 16

C = 2 3 9 10 11 12 15 16

TM

(1) Binary Tree :



2^{l-1}

$2^{1-1} = 1$

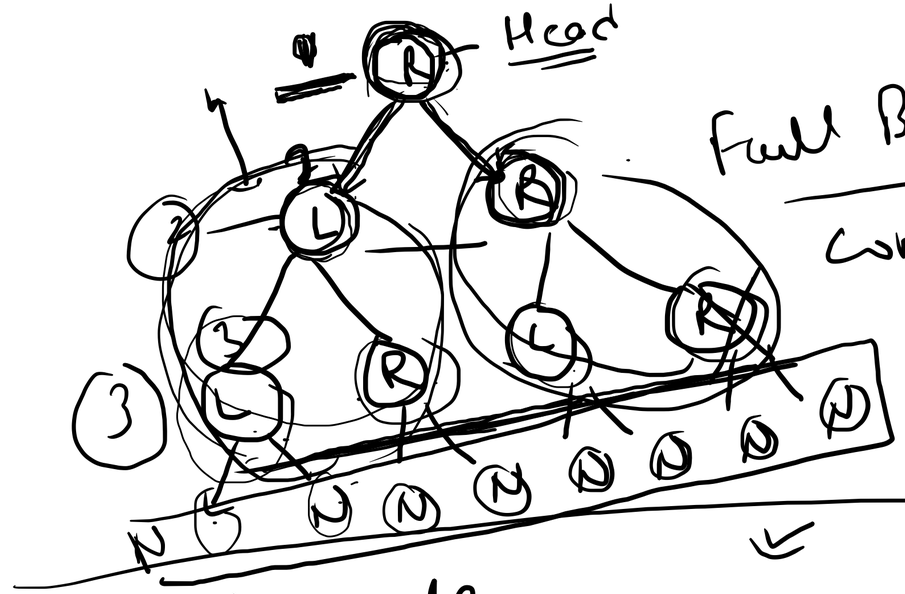
$2^{2-1} = 2$

$2^{3-1} = 4$

struct Node * Root;

Root = (struct Node*) malloc (sizeof(struct Node));

Root->data = 10;
Root->Left = NULL;
Root->Rt = NULL;



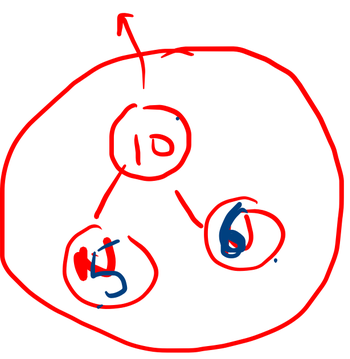
struct Node

{
int data;
struct Node* left;
struct Node* Right;
};

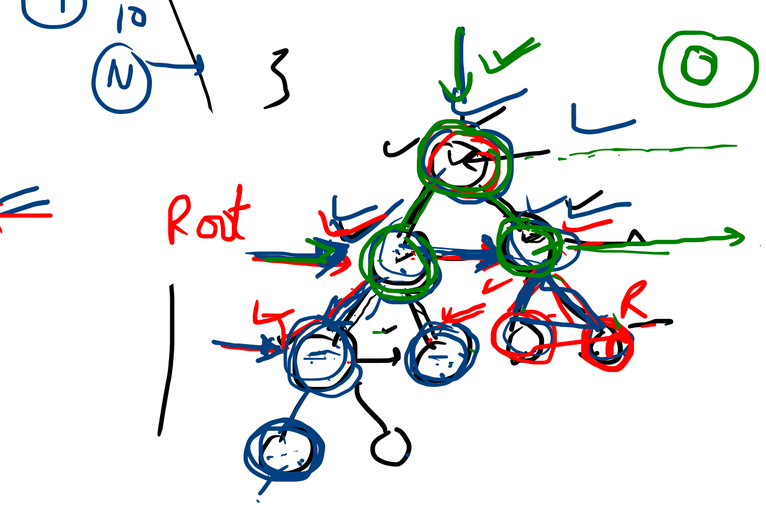
Create Binary Tree:

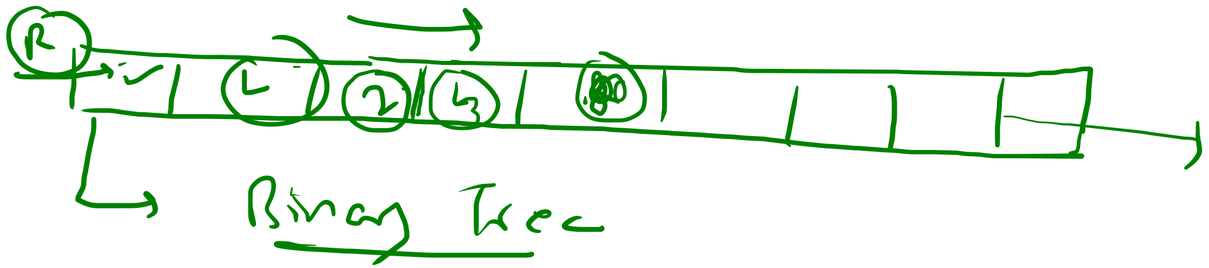
```
Typedef struct Node Node;

Node* NewNode(int x);
{
    Node* New = (Node*) malloc(sizeof(Node));
    New->Data = x;
    New->Left = NULL;
    New->Right = NULL;
    return New;
}
```

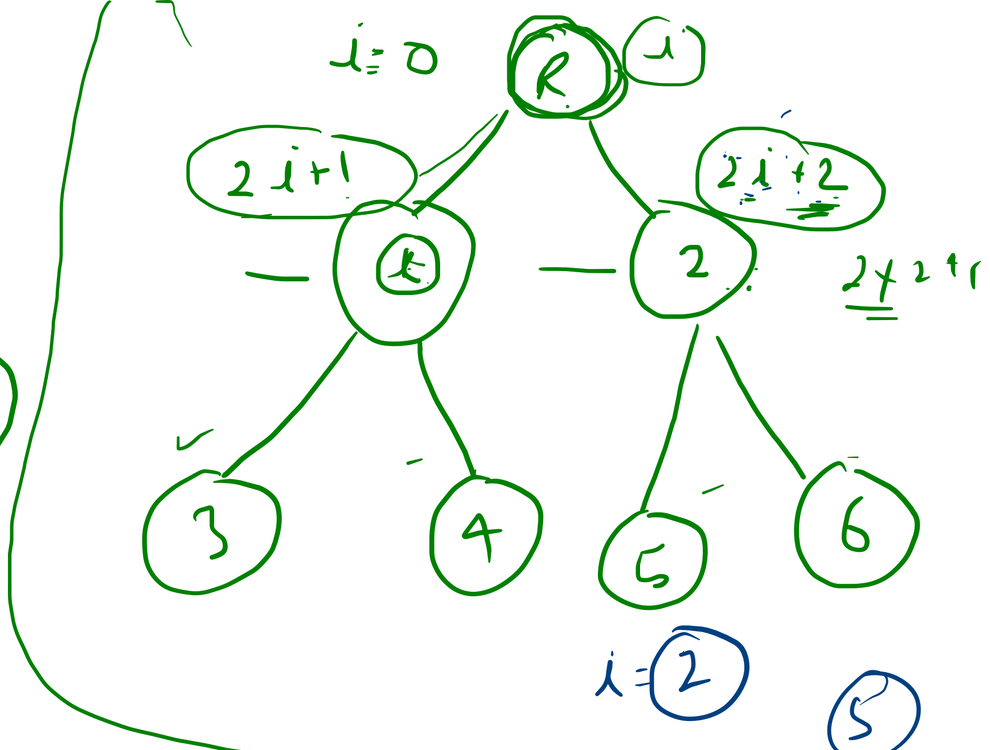
```
Node*  Create BT ( Node* Root  int x[ ], int N )
       {
          int i = 0 ;
          if ( Node = = NULL && N = = 1 )
          {
             Node * New = Create Node ( x[i++] );
             return New ;
          }
          else
          { Node * Root = Create Node ( x[i++] ) ;
            while ( i < N )
            {
               Root -> Left = Create Node ( x[i++] ) ;
               if ( i < N ) { Root -> Right = Create Node ( x[i++] ) ; }
               Root = Root -> Left ;
            }
          }
       }
```




```
Main
{ int x[ ] = {2, 3, 5, 10, 9};
  Node * Root = NULL ;
  int N = sizeof ( x )
  Root = Create BT ( Root, x, N )
```







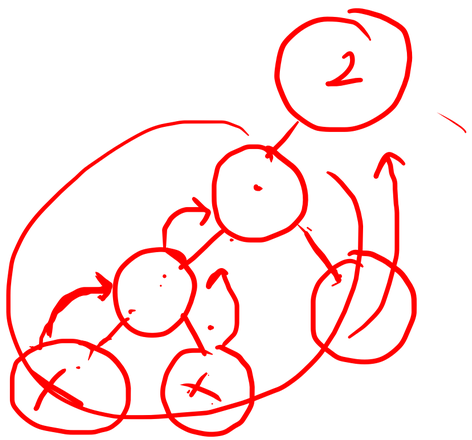

```
Node* InnerNode (int X[], Node* Root
                 int i, int N)
{ if (i < N)
  {
    Node* New = CreateNode (X[i]);
    Root = New;
    Root->Left = InnerNode (X, Root, 2i+1, N);
    Root->Right = InnerNode (X, Root, 2i+2, N);
  }
  return (Root);
}
```

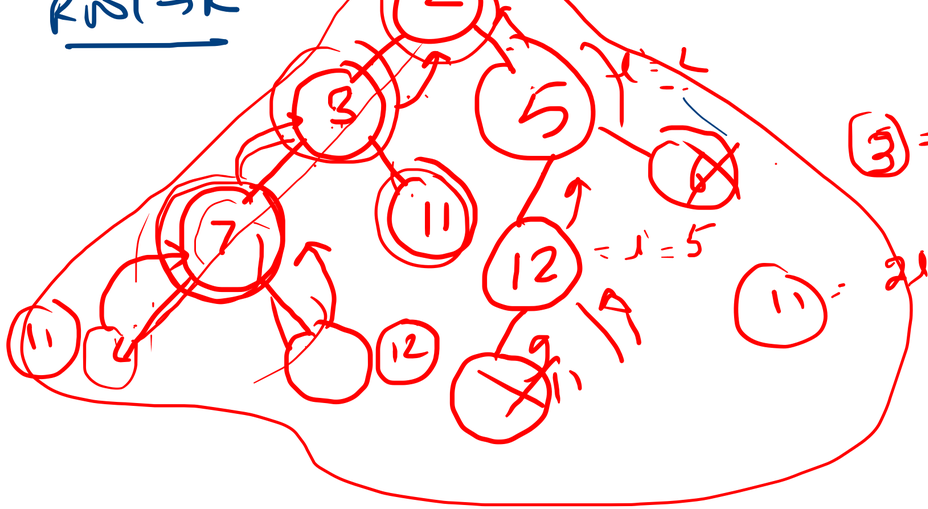
```
Main()
{
  int X[] = {2, 3, 5, 7, 11, 12 ...
  N = sizeof(X)
  Node* Root = NULL;
  Root = InnerNode (X, Root
                    0, N)
```

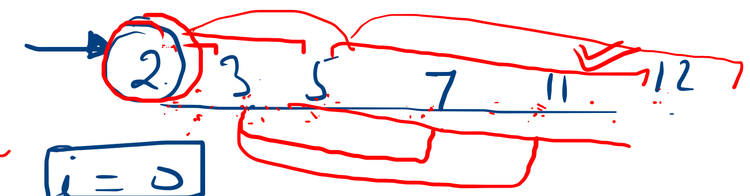
```
Node* InsertNode(int X[], int i, int N, Node* Root)
{
    if (i < N)
    {
        Node* New = CreateNode(X[i]);
        Root = New;
        Root->Left  = InsertNode(X, 2*i+1, N, Root->Left);
        Root->Right = InsertNode(X, 2*i+2, N, Root->Right);
    }
    return(Root);
}
```

```
int main()
{
    int X[] = {2,3,5,7,11,12};
    N = sizeof(X)/...
    Node* Root = NULL;
    Root = InsertNode(X, 0, N, Root);
```

i	N	Root	Root->L	Root->R
0	6	2	(X, 3, 1, N)	(X, 5, 2, N)
1	6	3	(X, 7, 3, N)	(X, 11, 4, N)
3	6	7	(X, 11, 7, N)	(X, 12, 8, N)
7	6	11	(X, 12, 15, N)	(X, 11, 16, N)
15	6	12	(X, 11, 31, N)	(X, 9, 32, N)
31	6	11	(X, 9, 63, N)	(X, 9, 64, N)







N=6

IN(X, NULL, 0, 6)